Simulation of VANET Routing Protocols

Jansen Lazaro
Supervised by Dr. Parimala Thulasiraman

Report on VANET Protocol Simulation and Comparison of AODV, DSR, and DSDV.

NS-3 with SUMO Simulation:

An experiment is necessary to evaluate VANET protocols and services. There are two types of experiment that can be done, an outdoor experiment and experiment using simulators. Before, we could implement VANET in real World, a set of experiments should be carried out to test it. These experiments are expensive and highly complex. There are many factors that have to be considered. One is routing protocols which is the main theme of this project. For this purpose, software simulations can be implemented to eradicate such factors.

To simulate VANET we will need two types of simulation for realism, Network simulation and Traffic simulation. Network simulators are used to evaluate the network protocols or say routing protocols for issues related to the communication among vehicles, and application in a variety of conditions whereas traffic simulators are used for Traffic engineering and transportation.

Capabilities

NS3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. SUMO is a microscopic vehicular traffic simulator. Capable of the following:

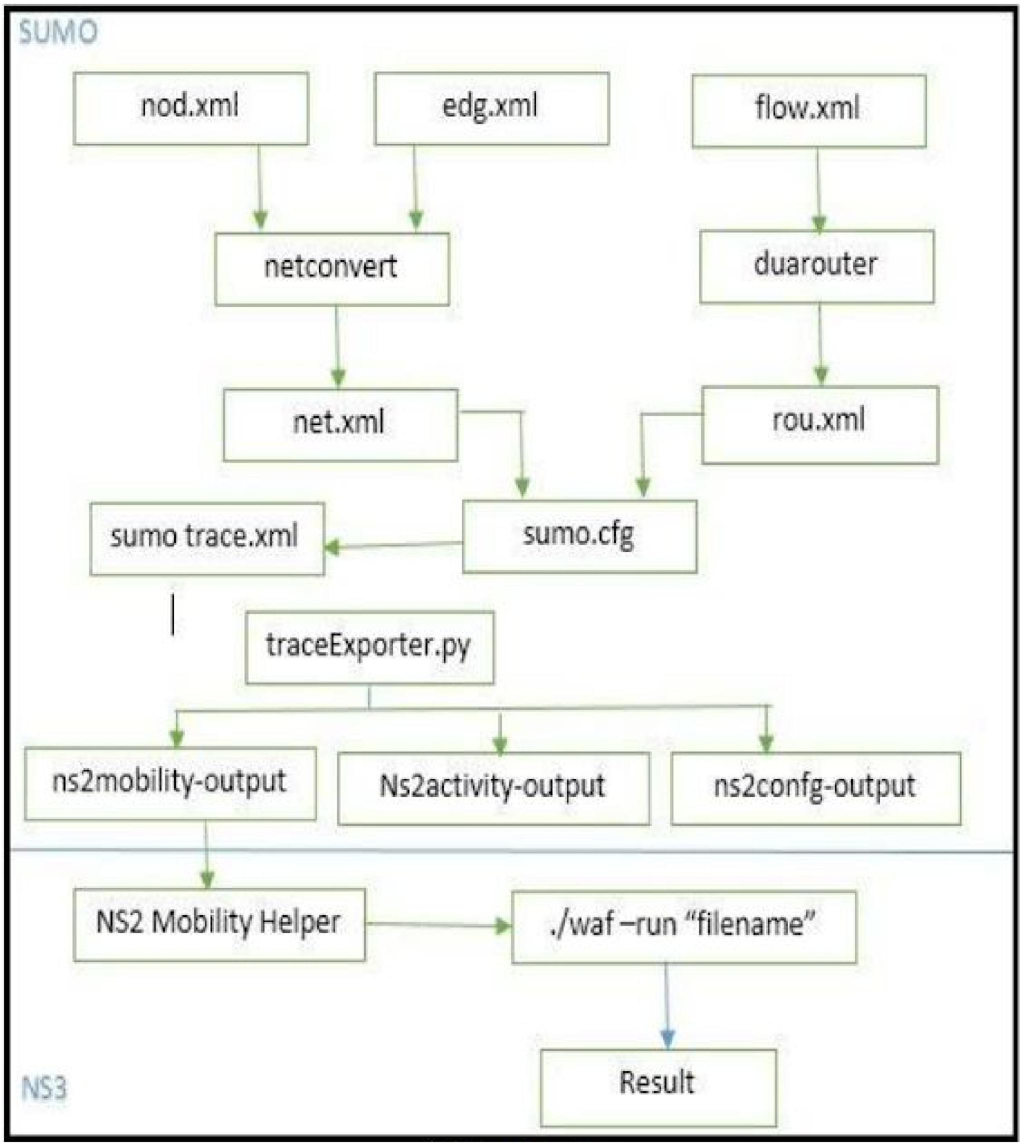
- Can model vehicles, pedestrians and public transport
- Can import maps, or generate custom road networks
- Can model car following models and inner junction traffic
- Gas emission, traffic light frequency, etc

Disadvantages

- Steep learning curve
- Can take a long time to configure since details have to be specified

- The user documentation consists mainly on its wiki
- SUMO architecture consists on a number of small programs which you may need to know and possibly configure and run each tool to maximize realism

The key reason for selecting SUMO as road simulation software lies in its strong capability of traffic scenario generation and operation.



Previously in the VANET industrial project, I implemented AODV in NS-3 using SUMO tcl file to imitate VANET topology and simulate the routing protocol.
However, the main goal of my project is to compare different protocols, in order to figure out it's differences in throughput, mean delay, bitrate, etc. As part of my industrial project at the UofM, I've simulated three different protocols AODV, DSR, and DSDV. The three protocols share the

same parameters and topology.

Network Protocols:

Network protocols are procedures that define communication between two or more devices over a network. Thus, for each protocols we need to setup a suitable network. In order to do this we have to understand what each protocols do.

AODV

Ad Hoc On-Demand Distance Vector Routing
By far the easiest, a reactive routing protocol. It creates routes on demand when source wants to send data to destination. It uses a destination sequence number which makes it different from other routing protocols. AODV handles route discovery with route request(RREQ) and route reply (RREP) messages. Uses sequence number to keep up-to-date.

DSR

Dynamic Source Routing also reactive.
The routes are stored in route cache. AODV and DSR have significant differences. In AODV when a node sends a data packet to destination node then data packet only contains the destination address. On the other hand in DSR the full routing information is contained in data packet which causes more routing overhead than AODV.

DSDV

Destination-Sequenced Distance Vector routing protocol is a pro-active.
Every node will maintain a table listing all the other nodes it has known either directly or through some neighbors. Every node has a single entry in the routing table. The entry will have information about the node's IP address, last known sequence number and the hop count to reach that node. Along with these details the table also keeps track of the nexthop neighbor to reach the destination node, the timestamp of the last update received for that node.

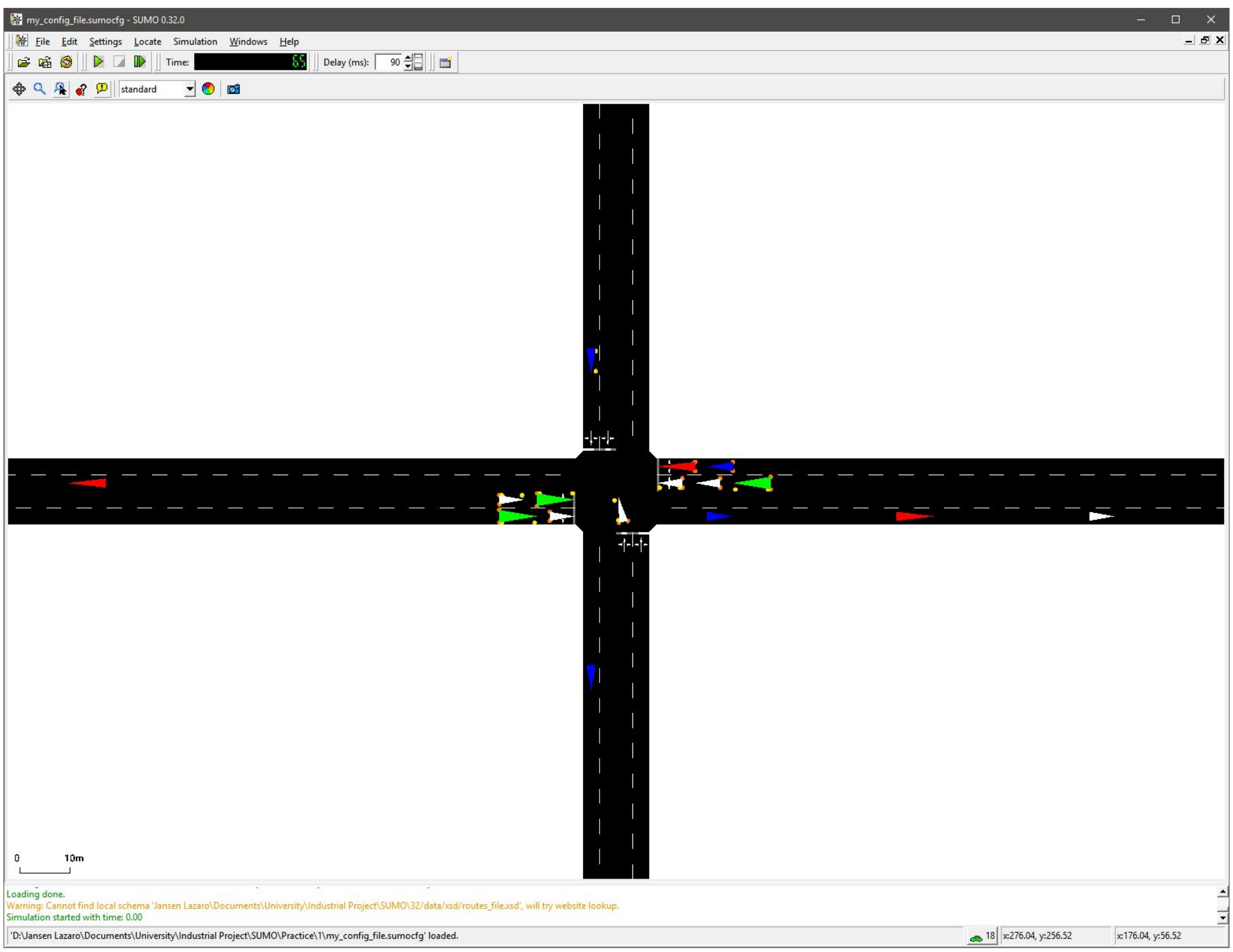
The Implementation:

First we define the parameters that will be universal for all our protocols.
For simplicity and easier implementation we will use the following:

Scenario area 200 m × 200 m
Communication range 50 m
MAC protocol IEEE 802.11 p
Simulation time 100 s
Vehicle velocity max 45 km/h
Bit rate 6 mb/s
Data packet size 1024 Bytes
Mobility ns2mobilitymodel

Creating a SUMO scenario

I made a simple 4-way intersection and set the max velocity to 45 km/h. The scenario area is set as 200m x 200m as specified above.



Using SUMO file in NS-3

We are able to use trace files in NS-3 that are produced in SUMO. This allows us to create a mobility in NS-3 that is otherwise complicated if using NS-3 alone. Generating trace files is a two step process.

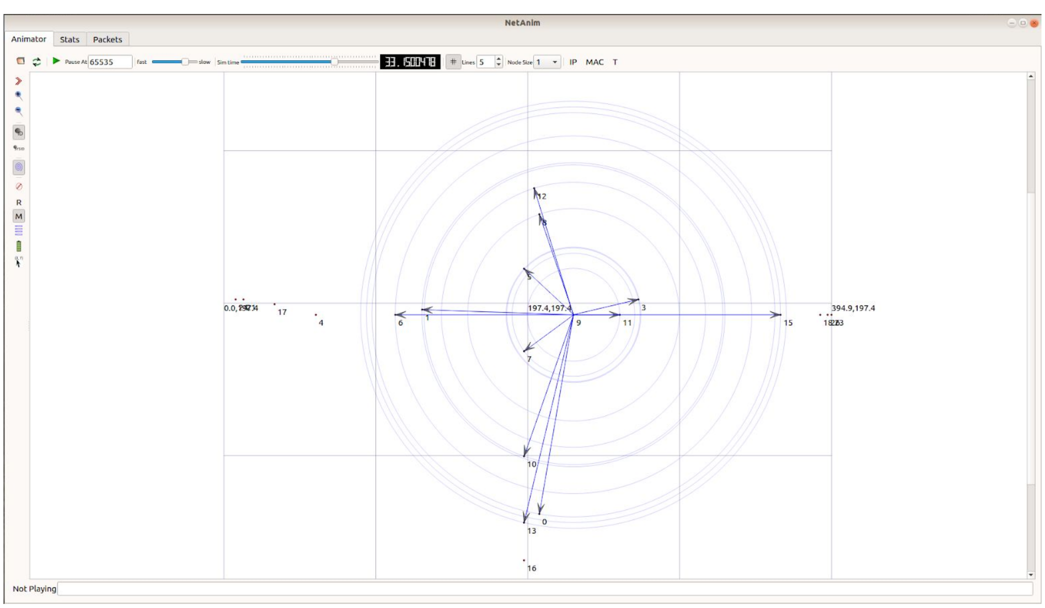
First we create a SUMO fcd output attribute by using this command:
"sumo-c My-scenario.sumo.cfg --fcd-output sumoTrace.xml"

Second, we convert the created SUMO trace file into actual trace file format. In our case, it is .tcl file which is NS2 trace file. There will be three output files: mobility file, activity file and config file.
"./traceExporter.py–fcd-input=smpleTrace.xml–ns2mobility-output=mobsmple.tcl"
"./traceExporter.py–fcd-input=smpleTrace.xml–ns2activity-output=actsmple.tcl"
"./traceExporter.py–fcd-input=smpleTrace.xml–ns2config-output=consmple.tcl"

Simulation

The biggest challenge was figuring out how to setup the VANET network environment. In the previous simulations, I used a MANET network environment for the three protocols. Such network will not work in VANET since the nodes are moving fast and packet loss is much more frequent.

Each protocol is implemented using the NS-3 modules. A node is created for each vehicles in the scenario, then a netdevice module is attached to the node where we specify what type of channel is used. For our simulation, its MAC protocol IEEE 802.11 p. Then we define the routing protocol, AODV. We then assign a base ip address for each node. Finally, we create a socket for each node to create a network.



Result

Using the flow monitor of NS-3 the result of the AODV is the following:

```
Flow Id:1
======
UDP 10.0.0.1/49153--->10.0.0.50/8080

Tx bitrate:8.50188kbps
Rx bitrate:8.50211kbps
Mean delay:11.8805ms
Packet Loss ratio:0%

timeFirstTxPacket=  1e+9ns
timeFirstRxPacket=  1.00424e+9ns
timeLastTxPacket=  9.9e+10ns
timeLastRxPacket=  9.90016e+10ns
delaySum=          1.17617e+9ns
jitterSum=         2.03208e+9ns
lastDelay=         1.17617e+9ns
txBytes=           104148
rxBytes=           104148
txPackets=         99
rxPackets=         99
lostPackets=       0
timesForwarded=    2

delayHistogram nBins:1014
Index:1 Start:0.001 Width:0.001 Count:96
Index:2 Start:0.002 Width:0.001 Count:1
Index:4 Start:0.004 Width:0.001 Count:1
Index:1013 Start:1.013 Width:0.001 Count:1

jitterHistogram nBins:1013
Index:0 Start:0 Width:0.001 Count:95
Index:2 Start:0.002 Width:0.001 Count:1
Index:1012 Start:1.012 Width:0.001 Count:2

packetSizeHistogram nBins:53
Index:52 Start:1040 Width:20 Count:99

flowInterruptionsHistogram nBins:8
Index:3 Start:0.75 Width:0.25 Count:45
Index:4 Start:1 Width:0.25 Count:51
Index:7 Start:1.75 Width:0.25 Count:1
```

```
Flow Id:2
======
UDP 10.0.0.12/654--->10.0.0.1/654

Tx bitrate:-3.84e+8kbps
Rx bitrate:-3.84e+8kbps
Mean delay:3006.72ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.60079e+10ns
timeLastTxPacket=  2.30011e+10ns
timeLastRxPacket=  2.60079e+10ns
delaySum=          3.00672e+9ns
jitterSum=         0ns
lastDelay=         3.00672e+9ns
txBytes=           48
rxBytes=           48
txPackets=         1
rxPackets=         1
lostPackets=       0
timesForwarded=    0

delayHistogram nBins:3007
Index:3006 Start:3.006 Width:0.001 Count:1

jitterHistogram nBins:0

packetSizeHistogram nBins:3
Index:2 Start:40 Width:20 Count:1

flowInterruptionsHistogram nBins:0
```

```
Flow Id:3
======
UDP 10.0.0.12/654--->10.0.0.50/654

Tx bitrate:-3.84e+8kbps
Rx bitrate:-3.84e+8kbps
Mean delay:11.772ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.30129e+10ns
timeLastTxPacket=  2.30011e+10ns
timeLastRxPacket=  2.30129e+10ns
delaySum=          1.1772e+7ns
jitterSum=         0ns
lastDelay=         1.1772e+7ns
txBytes=           48
rxBytes=           48
txPackets=         1
rxPackets=         1
lostPackets=       0
timesForwarded=    0

delayHistogram nBins:12
Index:11 Start:0.011 Width:0.001 Count:1

jitterHistogram nBins:0

packetSizeHistogram nBins:3
Index:2 Start:40 Width:20 Count:1

flowInterruptionsHistogram nBins:0
```

```
Flow Id:4
======
UDP 10.0.0.3/654--->10.0.0.1/654

Tx bitrate:0.0234618kbps
Rx bitrate:0.0251406kbps
Mean delay:1002.14ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.50052e+10ns
timeLastTxPacket=  5.30074e+10ns
timeLastRxPacket=  5.30077e+10ns
delaySum=          2.00428e+9ns
jitterSum=         2.00377e+9ns
lastDelay=         2.00428e+9ns
txBytes=           88
rxBytes=           88
txPackets=         2
rxPackets=         2
lostPackets=       0
timesForwarded=    0

delayHistogram nBins:2005
Index:0 Start:0 Width:0.001 Count:1
Index:2004 Start:2.004 Width:0.001 Count:1

jitterHistogram nBins:2004
Index:2003 Start:2.003 Width:0.001 Count:1

packetSizeHistogram nBins:3
Index:2 Start:40 Width:20 Count:2

flowInterruptionsHistogram nBins:113
Index:112 Start:28 Width:0.25 Count:1
```

```
Flow Id:5
======
UDP 10.0.0.3/654--->10.0.0.50/654

Tx bitrate:-3.84e+8kbps
Rx bitrate:-3.84e+8kbps
Mean delay:3002.97ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.60041e+10ns
timeLastTxPacket=  2.30011e+10ns
timeLastRxPacket=  2.60041e+10ns
delaySum=          3.00297e+9ns
```

```
Flow Id:6
======
UDP 10.0.0.19/654--->10.0.0.1/654

Tx bitrate:0.0234593kbps
Rx bitrate:0.0242791kbps
Mean delay:506.881ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.40146e+10ns
timeLastTxPacket=  5.30106e+10ns
timeLastRxPacket=  5.30108e+10ns
delaySum=          1.01376e+9ns
```

```
Flow Id:7
======
UDP 10.0.0.19/654--->10.0.0.50/654

Tx bitrate:-3.84e+8kbps
Rx bitrate:-3.84e+8kbps
Mean delay:18.4556ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.30196e+10ns
timeLastTxPacket=  2.30011e+10ns
timeLastRxPacket=  2.30196e+10ns
delaySum=          1.84556e+7ns
```

```
Flow Id:8
======
UDP 10.0.0.14/654--->10.0.0.1/654

Tx bitrate:0.0156418kbps
Rx bitrate:0.0156503kbps
Mean delay:12.4723ms
Packet Loss ratio:0%

timeFirstTxPacket=  2.30011e+10ns
timeFirstRxPacket=  2.30258e+10ns
timeLastTxPacket=  6.80088e+10ns
timeLastRxPacket=  6.80091e+10ns
delaySum=          2.49445e+7ns
```

Unfortunately, I was unable to recreate DSR and DSDV routing protocol in the VANET topology. I could not create the right network environment for both protocols to work in a VANET topology.

Summary:

This project is a simulation of routing protocols in a VANET environment. A lot of future work need to go into making this code ready for comparison. Learning these two technologies were a huge and significant part of my project. Both application presents a steep learning curve and undocumented modules. I’m creating a youtube tutorial for Dr. Parimala on how to use these technologies as a result of my industrial project. Although I didn’t reach my end result, I’m able to give something to Dr. Parimala to help her in her future research.